

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи освітнього ступеня «магістр»
за спеціальністю 121 «Інженерія програмного забезпечення»
(освітня програма «Інженерія програмного забезпечення»)
на тему:
«Аналіз продажів інтернет магазину»

Виконав студент групи ІПЗмп20-1
СТУПАКОВ Володимир Сергійович

Керівник роботи:
ВЛАСЕНКО Олег Васильович

Рецензент
ЄФРЕМОВ Юрій Миколайович

Житомир – 2024

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

«ЗАТВЕРДЖУЮ»

Зав. кафедри інженерії
програмного забезпечення
Тетяна ВАКАЛЮК
«26» червня 2024 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: Ступаков Володимир Сергійович
Тема роботи: «Аналіз продажів інтернет магазину»
Затверджена Наказом закладу вищої освіти від «26» червня 2024 р. № 174/ф
Вихідні дані для роботи: **предметом дослідження є використання сучасних технологій для аналізу продажів товарів. Об'єктом дослідження є сучасні методи та технології для ефективної розробки рекомендаційної системи.**
Консультанти з магістерської кваліфікаційної роботи із зазначенням розділів, що їх стосуються

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
1	Власенко О.В	20.07.2024	20.07.2024
2	Власенко О.В	20.07.2024	20.07.2024
3	Власенко О.В	20.07.2024	20.07.2024

Дата видачі завдання: «20» липня 2024р

РЕФЕРАТ

Кваліфікаційна робота магістра являє собою інтеграцію блоку з рекомендаціями в існуючий додаток для ведення електронної комерції та пояснювальної записки. Пояснювальна записка до кваліфікаційної роботи містить 58 сторінок, 30 ілюстрацій, 2 додатки та 16 літературних джерел.

Метою кваліфікаційної роботи є створення рекомендаційної системи з застосуванням алгоритму колаборативної фільтрації, написання програмного коду.

В роботі визначені головні завдання для розробки системи, проаналізовано методи вирішення поставленої задачі, обґрунтовано вибір інструментів для реалізації додатку та наведено методи які на сьогоднішній день активно використовуються для вирішення схожих задач.

Представлено алгоритм роботи програмного комплексу, структуру та взаємодію його компонентів.

КЛЮЧОВІ СЛОВА: КОЛАБОРАТИВНА ФІЛЬТРАЦІЯ, ДАНІ, АНАЛІЗ, МОДЕЛЬ, PYTHON, MONGODB.

ABSTRACT

The master's qualification work is the integration of a block with recommendations into an existing e-commerce application and an explanatory note. The explanatory note to the qualification paper contains 58 pages, 30 illustrations, 2 appendices and 16 literary sources.

The purpose of the qualification work is to get acquainted with the methods of data analysis and the construction of recommendation systems.

The work defines the main tasks for the development of the system, analyzes the methods of solving the given task, substantiates the choice of tools for implementing the application, and gives the methods that are actively used today to solve similar problems.

The algorithm of the software complex, the structure and interaction of its components are presented.

KEYWORDS: COLLABORATIVE FILTERING, DATA, ANALYSIS, MODEL, PYTHON, MONGODB.

ЗМІСТ

РЕФЕРАТ	3
ВСТУП	6
РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ.....	8
1.1. Обґрунтування актуальності теми	8
1.2. Аналіз існуючих сервісів та приклади аналізу продажів	11
1.3. Методи аналізу даних.....	14
1.4. Постановка задачі	21
Висновок до першого розділу:	22
РОЗДІЛ 2. РОЗРОБКА РЕКОМЕНДАЦІЙНОЇ СИСТЕМИ.....	24
2.1. Аналіз вхідних даних.....	24
2.2. Розробка рекомендаційної системи	30
Висновок до другого розділу:.....	34
РОЗДІЛ 3. ДОСЛІДЖЕННЯ ПРАЦЕЗДАТНОСТІ ВЕБ-ДОДАТКУ	35
3.1. Архітектура програмного комплексу та його розгортання	35
3.2. Оцінка та аналіз якості моделі.....	36
3.3. Перевірка працездатності додатку	40
Висновки до третього розділу:	45
ВИСНОВКИ.....	46
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	47
ДОДАТКИ.....	49

ВСТУП

Актуальність теми. Аналіз продажів є одним із ключових елементів управління інтернет-магазином. Світ електронної комерції розвивається з великою швидкістю, і компанії, що прагнуть залишатися конкурентоспроможними, змушені постійно адаптуватися до нових реалій ринку. Інтернет-магазини сьогодні мають доступ до величезних обсягів даних про своїх клієнтів, їхні вподобання та поведінку. У цьому контексті аналіз продажів виступає як один із ключових інструментів, що дозволяє компаніям:

- оцінити ефективність роботи магазину і менеджерів
- складати чіткі прогнози щодо процесів реалізації продукції тощо
- покращувати клієнтський досвід
- оптимізувати запаси
- підвищувати ефективність маркетингових кампаній
- ухвалювати рішення, що сприяють підвищенню показників продажів

Аналіз даних про продажі дозволяє інтернет-магазинам краще розуміти потреби та вподобання своїх клієнтів. Використовуючи ці знання, магазини можуть надавати персоналізовані рекомендації та пропозиції, що значно збільшує продажі та підвищує клієнтську лояльність та задоволеність.

Мета дослідження: створення рекомендаційної системи з застосуванням алгоритму колаборативної фільтрації, написання програмного коду.

Досягнення поставленої мети передбачає розв'язання таких **завдань**:

- Аналіз існуючих сервісів та розгляд прикладів аналізу даних.
- Аналіз вхідних даних.
- Реалізація рекомендаційної системи.
- Оцінка та аналіз якості рекомендацій.
- Перевірка працездатності додатку.

Об'єктом дослідження є сучасні методи та технології для ефективної розробки рекомендаційної системи.

Предметом дослідження є використання сучасних технологій для аналізу продажів товарів.

Методи дослідження. Концепція технології створення даної системи, опис розроблених технологій.

Наукова новизна. Дослідження використання рекомендаційних систем при аналізі продажів.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1. Обґрунтування актуальності теми

Аналіз продажів у роздрібній торгівлі є комплексом заходів, спрямованих на оцінку якості прийнятих рішень та усунення проблем у процесі реалізації з метою збільшення прибутку. Дані про продажі виступають ключовим матеріалом при оцінці ефективності роботи магазину [11].

Аналіз продажів є одним із ключових елементів управління інтернет-магазином. Світ електронної комерції розвивається з великою швидкістю, і компанії, що прагнуть залишатися конкурентоспроможними, змушені постійно адаптуватися до нових реалій ринку. Інтернет-магазини сьогодні мають доступ до величезних обсягів даних про своїх клієнтів, їхні вподобання та поведінку. У цьому контексті аналіз продажів виступає як один із ключових інструментів, що дозволяє компаніям:

- оцінити ефективність роботи магазину і менеджерів
- складати чіткі прогнози щодо процесів реалізації продукції тощо
- покращувати клієнтський досвід
- оптимізувати запаси
- підвищувати ефективність маркетингових кампаній
- ухвалювати рішення, що сприяють підвищенню показників продажів

Аналіз даних про продажі дозволяє інтернет-магазинам краще розуміти потреби та вподобання своїх клієнтів. Використовуючи ці знання, магазини можуть надавати персоналізовані рекомендації та пропозиції, що значно збільшує продажі та підвищує клієнтську лояльність та задоволеність.

Інтернет-магазини можуть виявляти найпопулярніші продукти та категорії, а також визначати, які товари найчастіше купуються разом. Це дозволяє оптимізувати асортимент та підготувати ефективніші акції. Крім того, аналіз відгуків клієнтів та оцінок товарів допомагає виявляти проблемні зони та покращувати якість обслуговування.

Визначення сезонних тенденцій дозволяє точно прогнозувати попит та планувати закупівлі. Це допомагає уникати як дефіциту товарів, так і надлишкових запасів, що знижує витрати на зберігання та логістику.

Аналізуючи продажі, інтернет магазини можуть оцінювати ефективність маркетингових кампаній та виявляти найефективніші канали та стратегії. Це дозволяє оптимізувати витрати на рекламу та збільшити повернення на інвестиції (ROI). Інтернет-магазини можуть швидко адаптувати свої стратегії на основі отриманих даних, що забезпечує їхню конкурентоспроможність.

Існує безліч методів аналізу продажів магазину за різними критеріями. Кожен окремий підхід дозволяє оцінити ефективність застосованих стратегій з різних аспектів. Розглянемо найпопулярніші [11]:

ABC аналіз - дозволяє класифікувати товари магазину за прибутковістю, використовуючи принцип Парето 80/20. Товари поділяються на три групи:

Група А: позиції, що приносять найбільший прибуток (80% прибутку від 20% асортименту).

Група В: товари зі стабільно середнім попитом.

Група С: товари з найменшою рентабельністю.

XYZ аналіз - оцінює рівномірність попиту, що є ключовим для оптимізації асортименту. Товари поділяються за коефіцієнтом варіації:

Група Х: товари з коефіцієнтом варіації 0-10%, що вказує на стабільний попит, змінюваний не більше ніж на 10% на місяць.

Група Y: товари з коефіцієнтом варіації 10-25%.

Група Z: товари з найменш передбачуваним попитом, з варіацією понад 25%.

Коливання попиту допомагають оцінити прибутковість кожної групи товарів та реалістичність планових показників продажів.

Визначення рентабельності - це показник прибутковості компанії, економічної ефективності вкладених коштів.

Аналіз динаміки продажів - оцінює зміни обсягу продажів (обороту) з часом. Він дозволяє порівняти показники поточного періоду з попередніми, щоб зрозуміти, наскільки змінилася виручка.

Аналіз виконання плану - допомагає визначити, наскільки магазин досяг запланованих показників продажів, тобто наскільки план було виконано або недовиконано.

Факторний аналіз продажів - дозволяє виокремити конкретні чинники, що впливають на реалізацію товарів.

Існує декілька ключових напрямів аналітики, які необхідно вивчати власникам інтернет-магазинів [12].

Трафік - власник сайту повинен знати обсяг трафіку та його характеристики. Аналітика виявить джерела трафіку та типи пристроїв, з яких користувачі відвідують інтернет-магазин. Наприклад, якщо значна частина клієнтів заходить зі смартфонів, варто оптимізувати мобільну версію сайту, зробивши її швидкою та зручною. Або, якщо відвідувачі, що перейшли за рекламою в Instagram купують більше чим відвідувачі з Facebook - потрібно зменшити бюджет на рекламу в Facebook на користь Instagram.

Аудиторія - аналізуючи дані про своїх клієнтів, включаючи їх вік, стать, соціальний статус, місцезнаходження тощо, дозволяє краще розуміти свою цільову аудиторію і ефективніше просувати свій магазин.

Поведінка - вивчаючи, як відвідувачі взаємодіють із сайтом, зокрема, як вони переміщуються по сторінках та на що натискають, наприклад, додавання товарів у кошик, ви зможете налаштувати динамічний ремаркетинг.

Результати просування - після запуску рекламної кампанії важливо коригувати її налаштування для досягнення кращих результатів. Аналітика покаже, наскільки ефективною є реклама, чи приносить вона прибуток і чи варто продовжувати використовувати цей канал просування. Докладніше про розрахунок ефективності реклами ми писали у нашій статті.

Результат торгівлі - інформація про найпопулярніші товари, бренди та категорії допоможе зрозуміти, в якому напрямку розвивати бізнес. Оцінюючи

частоту транзакцій та середній чек, ви зможете прогнозувати дохід та планувати подальші дії.

Чим більше даних з різних джерел аналізується, тим складніше їх об'єктивно оцінити. Візуалізація даних значно полегшує їх інтерпретацію та осмислення. У цьому може допомогти інструмент BI на платформі BES. Швидкий збір і обробка даних дозволяють зіставляти результати різних метрик. Перетворення даних аналізу в наочні дашборди допомагає оперативно приймати управлінські рішення [12].

У сучасному бізнес-середовищі, де конкуренція постійно зростає, аналіз продажів у роздрібній торгівлі відіграє важливу роль не лише у підрахунку основних економічних показників, але й у корекції внутрішніх процесів компанії та забезпечення стійкого зростання бізнесу. Чим детальніший аналіз продажів отримують співробітники, тим швидше зростає прибуток магазину. Інтернет-магазини, що активно використовують аналіз продажів, мають значну перевагу, що дозволяє їм адаптуватися до змін на ринку та задовольняти потреби клієнтів на найвищому рівні.

1.2. Аналіз існуючих сервісів та приклади аналізу продажів

Один з найяскравіших прикладів аналізу даних продажів є рекомендаційні системи. Вони стали невід'ємною частиною сучасної електронної комерції.

Задача забезпечення персоналізованих рекомендацій стає вкрай важливою у сучасній онлайн торгівлі. З еволюцією технологій і зростанням масивів даних, на ринку виникають численні рішення, що дають змогу користувачам отримувати персоналізовані пропозиції, базуючись на їхніх діях. Розглянемо найбільш відомі приклади використання рекомендаційних систем.

Netflix є одним із найвідоміших прикладів успішного використання рекомендаційних систем у сфері електронної комерції, зокрема, у галузі потокових відеосервісів. Netflix збирає величезні обсяги даних про поведінку користувачів. Це включає в себе історію переглядів, рейтинги, пошукові запити, час перегляду кожного шоу або фільму, а також взаємодії з платформою

(наприклад, додавання в список для перегляду). Після збору даних вони обробляються для виявлення патернів і трендів. Використовуються різні алгоритми машинного навчання для аналізу цих даних і розуміння вподобань кожного користувача.

Netflix застосовує колаборативну фільтрацію для рекомендації фільмів і серіалів. Їхній алгоритм враховує не лише рейтинги та перегляди користувачів, але й схожість між користувачами (user-based) та схожість між контентом (item-based).

Рекомендаційна система Netflix є ключовим елементом їхнього успіху, оскільки вона допомагає утримувати користувачів на платформі та збільшує час, проведений за переглядом контенту.

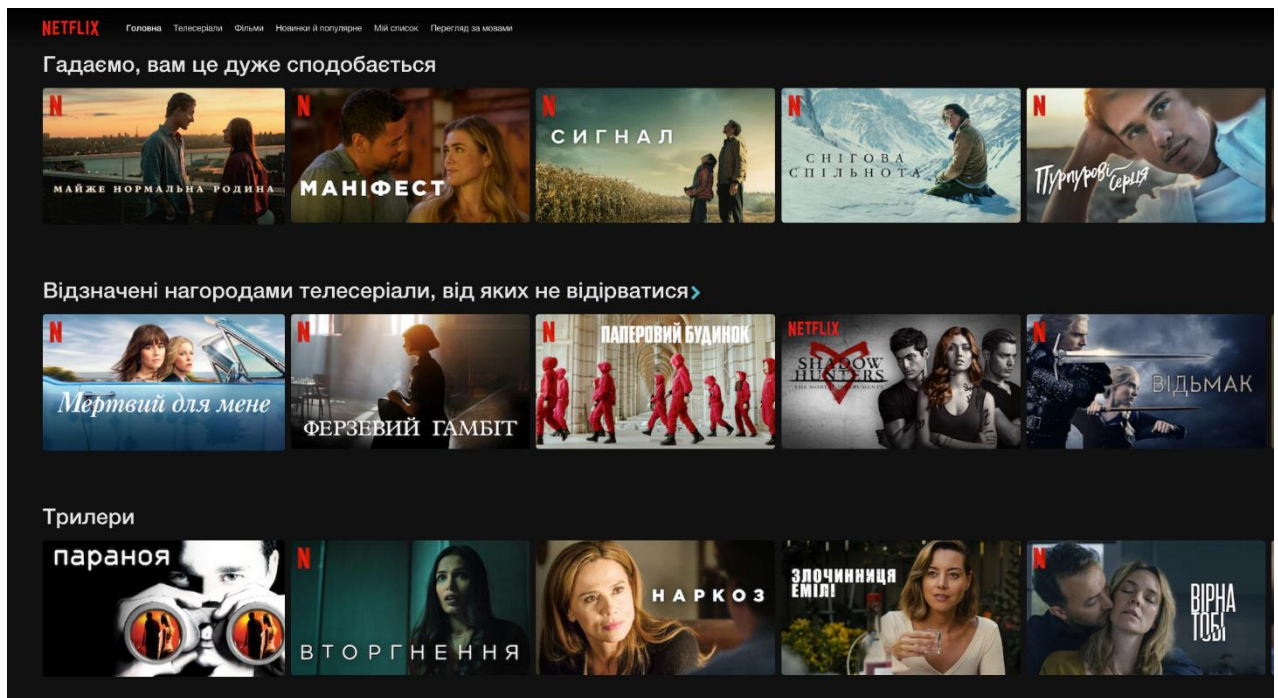


Рис. 1.1. Сервіс Netflix

Amazon, один із найбільших онлайн-ритейлерів у світі, застосовує один з найвідоміших і найуспішніших алгоритмів колаборативної фільтрації для надання персоналізованих рекомендацій.

Система збирає та аналізує історію покупок, перегляди товарів, пошукові запити, відгуки та рейтинги, а також дані про поведінку на сайті, такі як додавання товарів до кошика і списку бажань, що дозволяє пропонувати

продукти, які можуть зацікавити конкретного клієнта. Такий підхід виявився дуже ефективним і дозволяє Amazon значно збільшувати продажі завдяки точним рекомендаціям, підвищуючи ймовірність додаткових покупок.

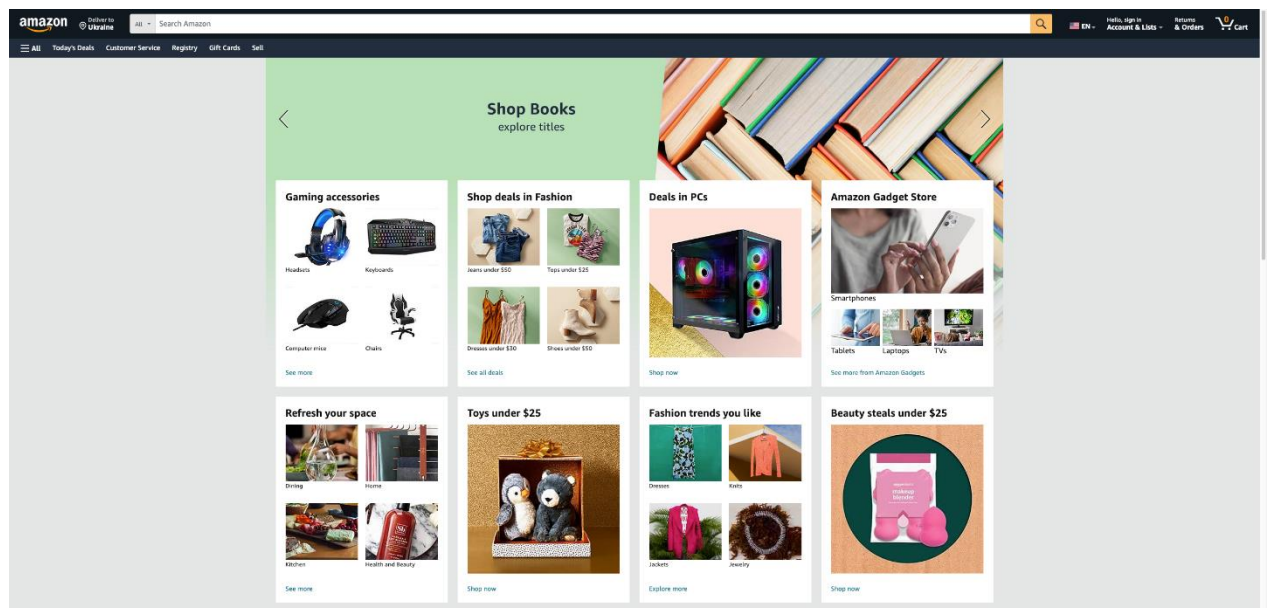


Рис. 1.2. Сервіс Amazon

Схожим на Amazon прикладом використання аналізу даних та рекомендаційних систем можуть бути такі сервіси, як Ebay, Etsy, Rozetka чи Prom.ua.

YouTube - платформа для відеохостингу, яка пропонує послуги зберігання та відтворення відео. Користувачі можуть завантажувати, переглядати, оцінювати, коментувати відео та інше. Сервіс застосовує два типи рекомендацій: неперсоналізовані і персоналізовані. Неперсоналізовані рекомендації відео це глобальні тренди, які є найбільш популярними в конкретному регіоні на даний момент. Персональні рекомендації, в свою чергу, засновані на попередніх діях користувача, таких як перегляди, лайки та коментарі. Система пропонує контент, який найбільше відповідає інтересам кожного глядача. Це сприяє збільшенню часу, який користувачі проводять на платформі, а також підвищенню рівня залученості та активної взаємодії з відео.

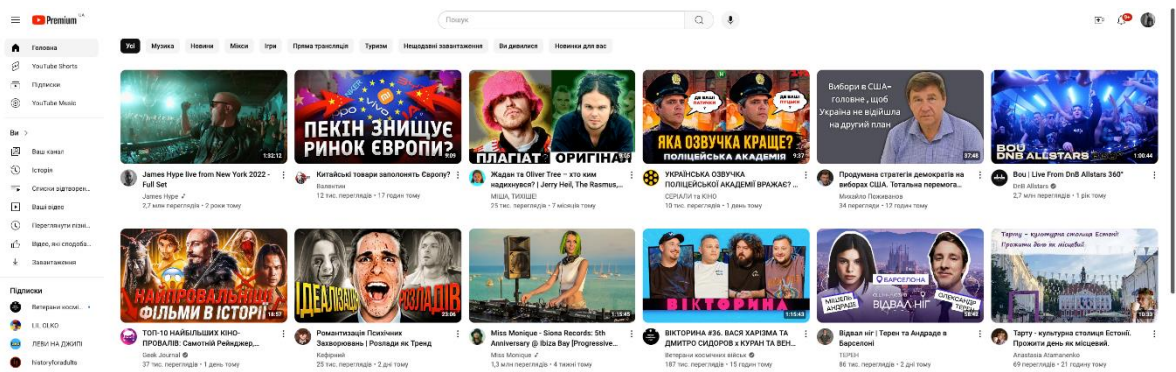


Рис. 1.3. Приклад персоналізованих рекомендацій на YouTube

Додатково, персоналізовані рекомендації допомагають користувачам відкривати новий контент, що забезпечує постійний потік цікавих відео та підтримує високий рівень задоволеності аудиторії.

1.3. Методи аналізу даних

Метою аналізу даних є здобуття знань про досліджуваний об'єкт - виявленні цінної інформації, формулюванні висновків і, зрештою, ухвалення зважених рішень. Аналіз даних може охоплювати різні аспекти та підходи, реалізовуватися за допомогою різноманітних інструментів — зокрема математичних, статистичних, машинного навчання та різних методів візуалізації, але головне його завдання — стиснення інформації.

Працюючи з відкритими даними, часто доводиться мати справу з великими обсягами інформації, що можуть включати сотні, тисячі або навіть сотні тисяч записів. Без спеціальних інструментів людині складно зробити з таких даних точні висновки. Тому необхідно використовувати різні засоби для узагальнення та консолідації інформації, представленої в "сирих" даних або мікроданих (рис. 1.4).

Статистика — наука збору, представлення, аналізу і розумної інтерпретації даних [13]. Статистичні методи можуть бути застосовані для отримання відповідей на такі питання:

- Які саме дані і в якій кількості потрібно зібрати?
- Як найкраще організувати та узагальнити дані?

- Як проаналізувати дані та зробити обґрунтовані висновки?
- Як оцінити достовірність висновків та визначити їхню невизначеність?

Описова статистика забезпечує інструменти для узагальнення та опису характеристик даних про продажі. Основні методи включають:

- Середнє значення: середнє значення продажів за певний період.
- Медіана: центральне значення в упорядкованому ряду даних.
- Мода: найчастіше зустрічається значення в наборі даних.
- Дисперсія та стандартне відхилення: вимірювання розсіювання даних навколо середнього значення.

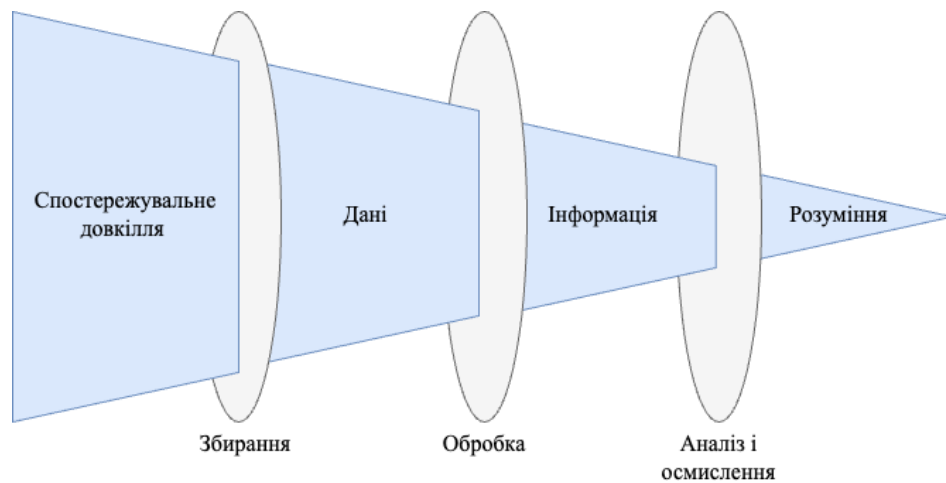


Рис. 1.4. Взаємодія даних, інформації і розуміння

Інфраструктурна статистика допомагає робити висновки про популяції на основі вибірок даних. Основні методи включають:

Регресійний аналіз: визначення взаємозв'язку між незалежними змінними (наприклад, рекламні витрати) та залежною змінною (наприклад, обсяг продажів).

Тестування гіпотез: визначення статистичної значущості відмінностей між групами даних.

Аналіз часових рядів: вивчення трендів продажів з плином часу та прогнозування майбутніх значень.

Описова аналітика використовує історичні дані для надання уявлень про минуле. Як приклад описової аналітики може виступати візуалізація даних: створення графіків та діаграм для наочного представлення даних.

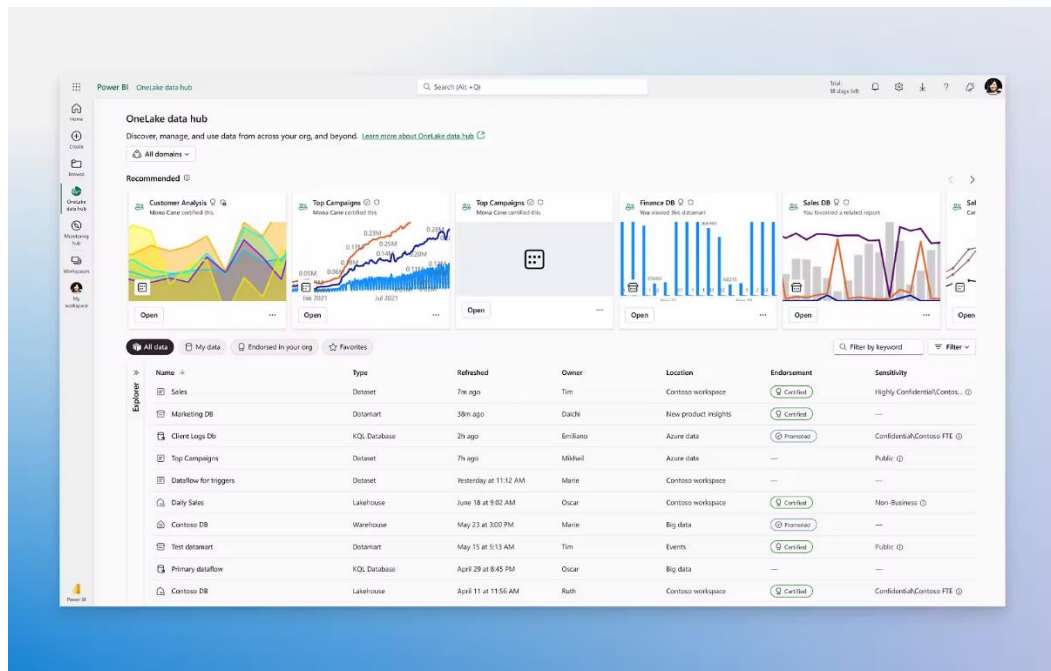


Рис. 1.5. Приклад візуалізації даних в Power BI

Діагностична аналітика спрямована на виявлення причин подій, що вже сталися. Методи включають:

- Аналіз кореляцій: виявлення зв'язків між різними змінними.
- Причинно-наслідковий аналіз: визначення причин змін у даних продажів.
- Прогностична аналітика використовує статистичні моделі та машинне навчання для прогнозування майбутніх подій. Основні методи включають:
- Моделювання часових рядів: прогнозування майбутніх значень на основі історичних даних.
- Регресія: передбачення кількісних значень на основі вхідних даних.

Окреме місце в області аналізу даних займають рекомендаційні системи.

Рекомендаційні системи спрямовані на передбачення уподобань користувачів та надання персоналізованих рекомендацій. Вони базуються на аналізі минулої поведінки користувачів, зібраних даних про їхні взаємодії з контентом та інших релевантних інформаційних джерел. Основна мета

рекомендаційних систем полягає у наданні корисних та релевантних рекомендацій, що задовольняють інтереси користувачів і сприяють їхньому задоволенню.

Виділяють три основні підходи: контентна фільтрація, колаборативна фільтрація та гібридний метод.

Контентна фільтрація (content-based filtering) базується на спільних атрибутах елементів контенту, які сподобалися користувачеві раніше. Вона позначає кожен елемент або користувача певними характеристиками, а потім вивчає та робить припущення щодо їхньої схожості [14].

Основна концепція полягає в тому, щоб пропонувати користувачеві елементи, базуючись на схожих атрибутах тих, які він уже вподобав. Наприклад, для рекомендацій фільмів можуть бути використані такі критерії, як жанр, режисер, акторський склад, рік випуску тощо. Ці параметри допомагають системі визначити, які елементи можуть бути схожими за змістом.

Цей тип рекомендаційних систем особливо ефективний у сферах, де важливий зміст та якісні характеристики елементів. Він дозволяє забезпечити користувачів релевантним та персоналізованим контентом, виходячи з їхніх інтересів та вподобань.

Також, існує інший метод контентної фільтрації - семантичний аналіз. Це використання методів обробки природної мови (NLP) для аналізу текстових описів товарів та знаходження схожих за змістом елементів.

Серед основних переваг систем рекомендацій на основі контенту можна виділити:

Персоналізація: Ці системи пропонують високий рівень персоналізованих рекомендацій, оскільки враховують контент, який користувач уже переглянув і оцінив.

Створення релевантних рекомендацій для нових користувачів: Вони ефективно працюють навіть для нових користувачів, які ще не мають значної історії взаємодій.

Пропозиція нішевого контенту: Такі системи сприяють відкриттю нішевого або маловідомого контенту, який може бути важко знайти іншими методами.

Однак, ці системи мають і свої обмеження. Вони враховують лише контент і не аналізують взаємодії користувачів, що може обмежувати їхню ефективність. Крім того, їм важко рекомендувати новий контент, оскільки вони базуються на вже відомих атрибутах. У таких випадках варто звернути увагу на системи на основі колаборативної фільтрації, які можуть краще впоратися з подібними завданнями.

Колаборативна фільтрація (collaborative filtering) є одним з найпоширеніших методів рекомендаційних систем та базується на припущенні, що користувачі, які мали схожі уподобання в минулому, ймовірно, матимуть схожі уподобання і в майбутньому. Колаборативна фільтрація може бути реалізована двома основними способами:

Методи пам'яті: ці методи використовують всю матрицю уподобань користувачів для знаходження схожих користувачів або товарів. Основні підходи включають алгоритми К-найближчих сусідів (KNN) та методи на основі косинусної подібності.

Методи моделювання: ці методи будують модель на основі матриці уподобань користувачів. Основні підходи включають матричну факторизацію (наприклад, SVD) та латентні факторні моделі.

Колаборативний фільтр можна поділити на два типи:

Item-Item: Цей метод аналізує схожість між елементами контенту. Якщо користувач позитивно оцінив певний елемент, система рекомендує інші схожі елементи, які також можуть йому сподобатися.

User-Item: Цей метод аналізує взаємодії між користувачами та контентом. Якщо два користувачі мають схожі вподобання, система рекомендує одному користувачеві елементи, які інший користувач оцінив позитивно, і навпаки.

Таким чином, фільтр Item-Item аналізує схожість контентних елементів, а фільтр User-Item – взаємодії, спираючись на схожі користувацькі вподобання.

Для надання рекомендацій на основі колаборативної фільтрації система використовує матрицю взаємодій, де кожен рядок представляє користувача, а кожен стовпчик – елемент контенту. Значення в матриці вказують на взаємодії між користувачами та елементами

Таблиця 1.1.

Приклад матриці взаємодії

	Item 1	Item 2	Item 3	Item 4	Item 5	Item 6
User 1	5	4	5			
User 2	4		5			
User 3		3	5		4	
User 4				3	4	
User 5			4	2	4	
User 6	3					5

Основні переваги системи на основі колаборативної фільтрації включають:

- Здатність до рекомендацій нових елементів: Система може пропонувати нові елементи, спираючись на взаємодії інших користувачів, навіть якщо ці елементи раніше не були відомі.
- Соціальний контекст: Врахування соціальних взаємодій та відносин між користувачами покращує якість рекомендацій, роблячи їх більш персоналізованими.
- Ефективність в умовах обмеженої інформації: Система добре працює, коли важко категоризувати елементи контенту, оскільки вона орієнтується на фактичні взаємодії користувачів.

Однак, системи на основі колаборативної фільтрації мають і свої обмеження. Одне з них – проблема «початкового холодного старту» для нових користувачів, які ще не мають історії взаємодій, що ускладнює надання релевантних рекомендацій на початковому етапі.

Гібридний підхід (hybrid filtering).

Гібридні методи комбінують колаборативну та контентну фільтрацію для підвищення точності рекомендацій. Основні підходи включають:

Комбіновані моделі: об'єднання результатів різних моделей для надання кращих рекомендацій.

Каскадні методи: використання одного методу для попереднього відбору рекомендацій, а іншого – для їхньої уточнення.

Методи мета-рівня: навчання моделі, яка визначає, який метод рекомендувати в конкретній ситуації.

В системах колаборативної фільтрації заснованих на роботі з пам'яттю необхідно визначити функцію схожості між двома користувачами. Існує багато функцій, які можуть бути використані для визначення схожості. Розглянемо дві основні формули: косинусну схожість та коефіцієнт кореляції Пірсона.

При використанні косинусної схожості близькість двох користувачів обчислюється як косинус кута між векторами, що відповідають їхнім рядкам у матриці оцінок. Така міра забезпечує найбільшу близькість, якщо оцінки користувачів щодо відповідних об'єктів пропорційні.

Формула для обчислення косинусної схожості:

$$sim(u, v) = \frac{\sum_{i \in I} r_{u,i} r_{v,i}}{\sqrt{\sum_{i \in I} r_{u,i}^2} \sqrt{\sum_{i \in I} r_{v,i}^2}} \quad (1.1)$$

де: $r_{u,i}$ та $r_{v,i}$ – оцінки користувачів u та v для об'єкта i .

Коефіцієнт кореляції Пірсона відображає ступінь лінійної залежності між двома центрованими векторами. Близькість залежить від того, наскільки високо або низько користувач оцінив товар відносно своїх інших оцінок. Для векторів користувачів u і v формула коефіцієнта кореляції має такий вигляд:

$$sim(u, v) = \frac{\sum_{i \in I} (r_{u,i} - \bar{r}_u)(r_{v,i} - \bar{r}_v)}{\sqrt{\sum_{i \in I} (r_{u,i} - \bar{r}_u)^2} \sqrt{\sum_{i \in I} (r_{v,i} - \bar{r}_v)^2}} \quad (1.2)$$

де: $r_{u,i}$ та $r_{v,i}$ – оцінки користувачів u та v для об'єкта i .

1.4. Постановка задачі

Мета кваліфікаційної роботи полягає у створенні ефективної рекомендаційної системи для продажу товарів, яка базується на колаборативній фільтрації. Ця система має надавати користувачам персоналізовані пропозиції на основі попередніх покупок користувача, що підвищують їхню задоволеність і покращують показники продажів.

Для досягнення цієї мети необхідно виконати такі завдання:

- Аналіз основних принципів та технологій колаборативної фільтрації;
- Вивчення існуючих методів та алгоритмів колаборативної фільтрації.
- Аналіз переваг та недоліків різних підходів до колаборативної фільтрації.
- Розробка алгоритму колаборативної фільтрації;
- Вибір підходу для колаборативної фільтрації (на основі об'єктів).
- Реалізація алгоритму колаборативної фільтрації з використанням метрики косинусної відстані.

Щоб розробити систему рекомендацій на основі історії покупок користувача, буде використано колаборативне фільтрування.

Ось кроки, які потрібно виконати, щоб розробити та розгорнути систему рекомендацій на основі історії покупок користувача [15]:

Формування даних. Множина користувачів U і множина товарів I .

Перетворення даних у матрицю «користувач-елемент»: у цій матриці деякий користувач $u \in U$ купує продукт товар $i \in I$. Кожен запис у матриці представляє історію покупок користувача для товару $r_{u,i}$ та може приймати значення 0 або 1. Наприклад, якщо користувач 1 придбав товари А і В, то запис у першому рядку та перших двох стовпцях матриці буде 1.

Обчислення подібності між користувачами: використаємо метод косинусної подібності (формула 1). Косинусна подібність вимірює кут між двома

векторами у багатовимірному просторі. У цьому випадку вектори представляють історію покупок двох користувачів.

Знаходження схожих користувачів: після обчислення подібності між користувачами потрібно знайти користувачів, які мають схожу історію покупок із цільовим користувачем. Чим більше схожі користувачі, тим більша вага їхніх рекомендацій.

Рекомендація товарів: коли схожі користувачі знайдені, мають бути рекомендовані товари, які вони придбали, але цільовий користувач ні. Товари можна сортувати за їхнім рекомендаційним балом, який обчислюється шляхом підсумовування ваги схожих користувачів, які придбали товар.

Розгортання системи рекомендацій. Розгорнемо систему рекомендацій за допомогою веб-додатку, RESTful API. Система буде робити запит до бази даних та отримувати історію покупок користувача та повертати список рекомендованих товарів.

Висновок до першого розділу:

В цьому розділі було проаналізовано актуальність теми аналізу даних електронної комерції та застосування рекомендаційних систем. Розглянуто важливість аналізу продажів у роздрібній торгівлі, зокрема для інтернет-магазинів. Аналіз продажів дозволяє оцінювати ефективність роботи магазину, прогнозувати попит, оптимізувати запаси, покращувати клієнтський досвід, підвищувати ефективність маркетингових кампаній та ухвалювати стратегічні рішення. Завдяки аналізу даних про продажі, магазини можуть краще розуміти потреби клієнтів, надавати персоналізовані рекомендації, виявляти популярні товари та проблемні зони, а також ефективніше планувати закупівлі. У сучасному конкурентному середовищі це забезпечує стійке зростання бізнесу та задоволення потреб клієнтів на високому рівні.

Також, було розглянуто найбільш відомі приклади використання рекомендаційних систем на прикладі сервісів Netflix, Amazon. Розглянуто різні методи аналізу даних, описано, що таке рекомендаційна система та як вона

працює, розглянуто різні підходи до методу колаборативної фільтрації, такі як Item-Item, User-user та гібридні, розглянуто функції схожості: косинусна та коефіцієнт Пірсона.

Поставлено задачу для кваліфікаційної роботи, що полягає у створенні ефективної рекомендаційної системи для продажу товарів.

РОЗДІЛ 2. РОЗРОБКА РЕКОМЕНДАЦІЙНОЇ СИСТЕМИ

2.1. Аналіз вхідних даних

Аналіз вхідних даних є ключовим етапом у процесі машинного навчання та аналітики. Цей етап включає дослідження, очищення та перетворення даних, які будуть використані для побудови моделей. Мета аналізу вхідних даних - отримати глибоке розуміння структури даних, а також виявити можливі проблеми, такі як пропущені значення, аномалії та некоректні формати.

Основними завданнями цього розділу є:

- Опис та візуалізація даних: Проведення огляду основних характеристик набору даних.
- Попередня обробка даних: Очищення даних від шуму, обробка пропущених значень.
- Перетворення даних: Визначення найбільш важливих змінних, перетворення даних у формат, придатний для аналізу та моделювання та імпорт в існуючу базу даних.
- Ретельний аналіз вхідних даних дозволяє побудувати більш точні та надійні моделі, які можуть забезпечити високий рівень прогнозування та прийняття рішень.

Тому важливо приділити достатньо уваги цьому етапу, щоб гарантувати успішність подальшого процесу машинного навчання.

Для збору даних для аналізу та машинного навчання існує багато джерел, таких як публічні датасети та спеціалізовані платформи. Ось кілька основних джерел, де зазвичай можна знайти дані:

- Kaggle - це онлайн-платформа для змагань з аналізу даних і машинного навчання, яка також має велику бібліотеку наборів даних
- UCI Machine Learning Repository - є одним з найстаріших та найпопулярніших ресурсів для дослідників у галузі машинного навчання.
- Google Dataset Search - пошукова система для наборів даних.

- Open Data on AWS - безкоштовні набори даних на платформі Amazon Web Services.

Вхідні дані для аналізу, а саме csv файл з даними про покупки користувачів магазину електроніки, були завантажені з ресурсу Kaggle. Переглянемо отриманий набір даних.

```
data = pd.read_csv('./drive/MyDrive/df1.csv')
data.info()
data.head()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 186379 entries, 0 to 186378
Data columns (total 3 columns):
#   Column                Non-Null Count  Dtype
---  -
0   Purchase Address      186379 non-null object
1   Product               186379 non-null object
2   Order ID              186379 non-null int64
dtypes: int64(1), object(2)
memory usage: 4.3+ MB
```

	Purchase Address	Product	Order ID
0	136 Church St, New York City, NY 10001	Macbook Pro Laptop	295665
1	562 2nd St, New York City, NY 10001	LG Washing Machine	295666
2	277 Main St, New York City, NY 10001	USB-C Charging Cable	295667
3	410 6th St, San Francisco, CA 94016	27in FHD Monitor	295668
4	43 Hill St, Atlanta, GA 30301	USB-C Charging Cable	295669

Рис. 2.1. Перегляд набору даних

Маємо набір даних, що містить більше 180 тисяч записів про покупки, та містить:

Стовпець «Purchase Address»: Адреса, на яку було оформлена покупка.

Стовпець «Product»: Назва продукту.

Стовпець «Order ID»: Ідентифікатор замовлення.

Для аналізу даних та побудови рекомендаційної системи цих даних буде достатньо.

Першочергово потрібно очистити данні від пропущених або некоректних значень. Видалимо рядки, де є пропущені значення.

```
data_cleaned = data.dropna()
missing_values = data_cleaned.isnull().sum()
print(missing_values)
```

⇒ Purchase Address	0
Product	0
Order ID	0

Рис. 2.2. Видалення рядків, де є пропущені значення

За допомогою бібліотеки `matplotlib` виведемо рейтинг продуктів по кількості замовлень. З графіку можемо бачити, що більш дешеві продукти та ті, що часто використовуються в побуті, наприклад зарядний пристрій, батарейки або навушники, очевидно, мають більшу кількість замовлень.

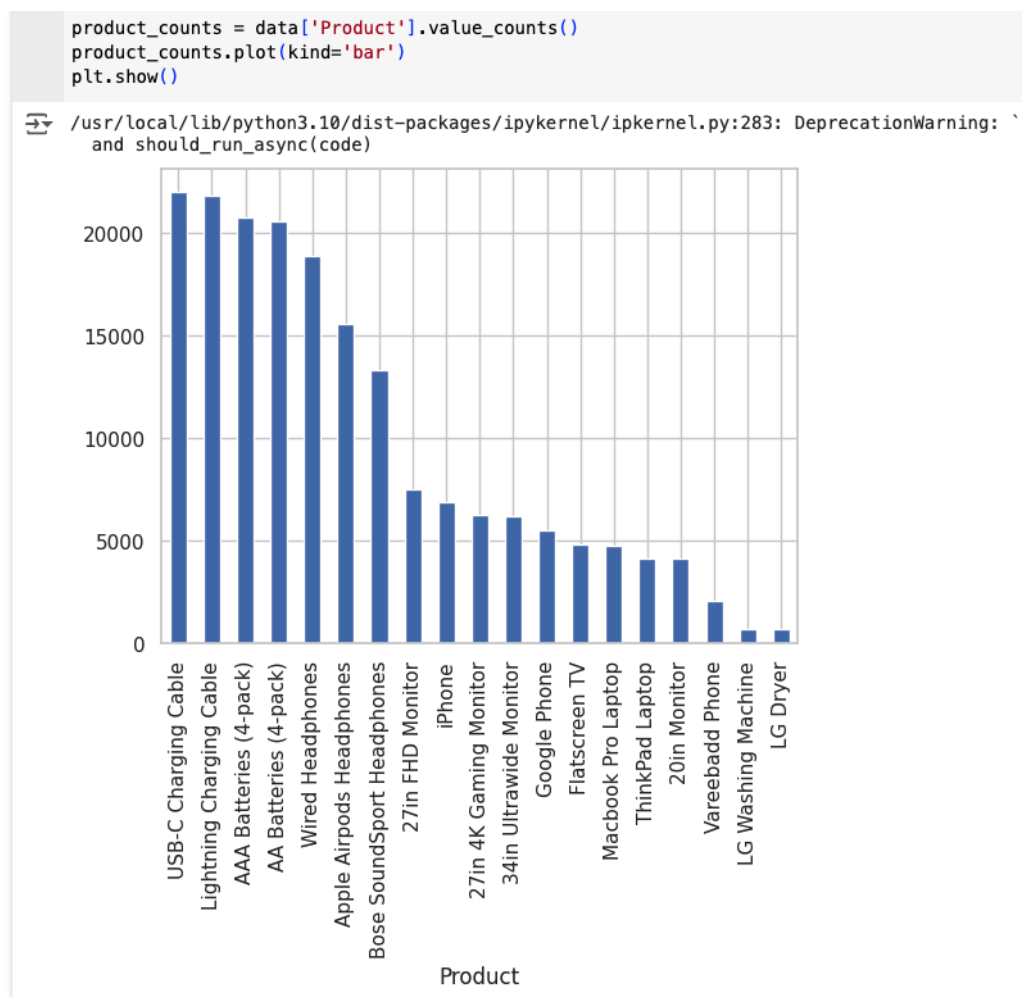


Рис. 2.3. Кількість замовлень продуктів

Далі, знайдемо продукти, що часто купують разом. Для цього, створимо бінарну матрицю для кожного замовлення і продукту. Функція `crosstab` створює перехресну таблицю, де рядки представляють замовлення, а стовпці — продукти. Значення в матриці будуть 1, якщо продукт був у замовленні, і 0, якщо не був. Далі, за допомогою “`np.dot`” обчислюємо матрицю спільної появи, де кожен елемент представляє кількість разів, коли два продукти з'являлися в одному замовленні.

```
product_matrix = pd.crosstab(index=data['Order ID'], columns=data['Product'])
co_occurrence_matrix = np.dot(product_matrix.T, product_matrix)
np.fill_diagonal(co_occurrence_matrix, 0)
co_occurrence_df = pd.DataFrame(co_occurrence_matrix, index=product_matrix.columns, columns=product_matrix.columns)
plt.figure(figsize=(10, 8))
sns.heatmap(co_occurrence_df, cmap="YlGnBu", annot=True, fmt='d')
plt.title('Product Co-Occurrence Matrix')
plt.show()
```

Рис. 2.4. Обчислення спільної появи продуктів

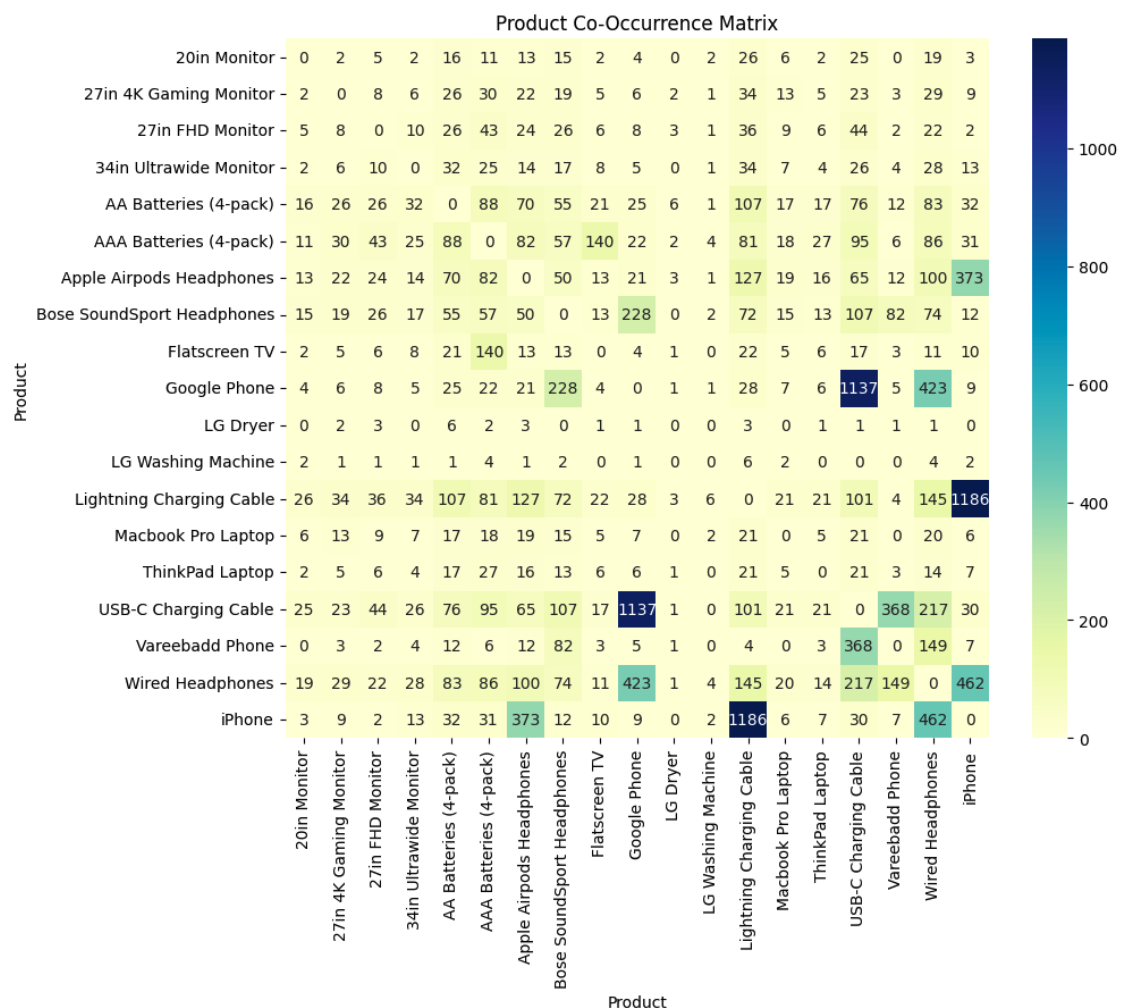


Рис. 2.4. Теплова карта спільної появи продуктів в одному замовленні.

З теплової карти чітко видно, що продукти iPhone та Lightning Charging Cable купують разом найчастіше. Також, можемо виділити пари, що часто зустрічаються разом, це: Google Phone та USB-C Charging Cable, iPhone та Apple AirPods Headphones та інші.

Можемо використовувати ці дані, як орієнтир для розробки та перевірки алгоритму рекомендаційної системи, так як ми чітко бачимо, що при купівлі iPhone ми можемо рекомендувати користувачу Lightning Charging Cable, Apple AirPods Headphones, Wired Headphones, так як ці продукти купують найчастіше разом.

Іншим варіантом відображення продуктів, хоча і не таким інформативним, може виступати граф.

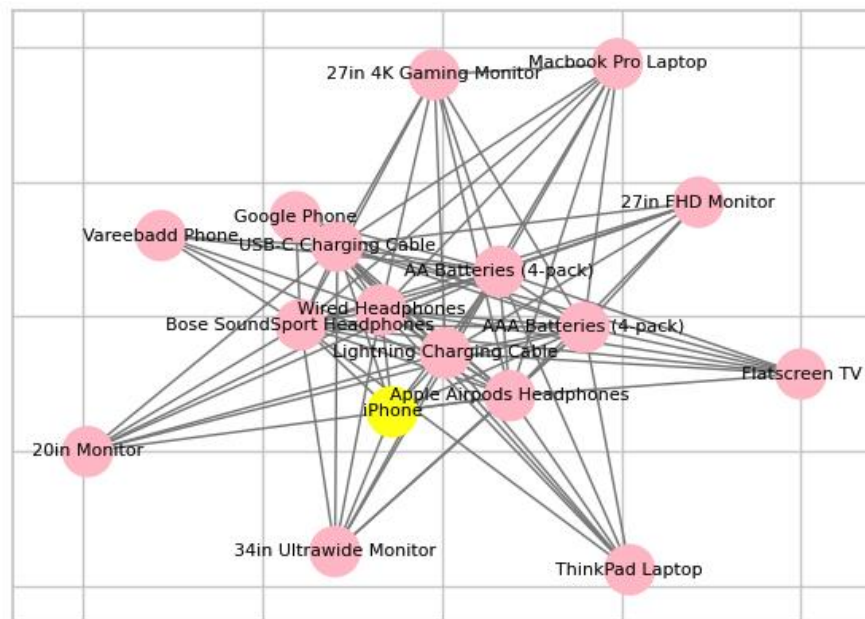


Рис. 2.5. Відображення спільної появи продуктів в одному замовленні у вигляді графа за допомогою бібліотеки networkx.

Наступним кроком, нам потрібно імпортувати завантажені дані в існуючу базу даних інтернет магазину. Для цього, потрібно модифікувати і привести до потрібного вигляду завантажені дані, а саме:

Вважатимемо, що замовлення на одну і ту саму адресу були зроблені одним і тим же користувачем, а тому стовпець «Purchase Address» можемо

замінити на “userId”, та кожне унікальне значення стовпця замінити на унікальний ідентифікатор існуючого користувача в базі даних.

Виділити кожен унікальний продукт в стовпці «Product» та додати як окремий продукт в базу даних, записавши в стовпчик «productId» ідентифікатор продукту з бази даних замість назви.

Завантажити всі замовлення з отриманого документу в базу даних.

База даних вже існує та була реалізована на Mongo DB. MongoDB — це документно-орієнтована база даних NoSQL, яка зберігає дані у форматі JSON-подібних документів із гнучкою схемою, тобто зберігає дані у колекціях, а не в таблицях, що дозволяє легко зберігати та керувати різними типами даних. Тому, структура сутності “Замовлення” змінилась.

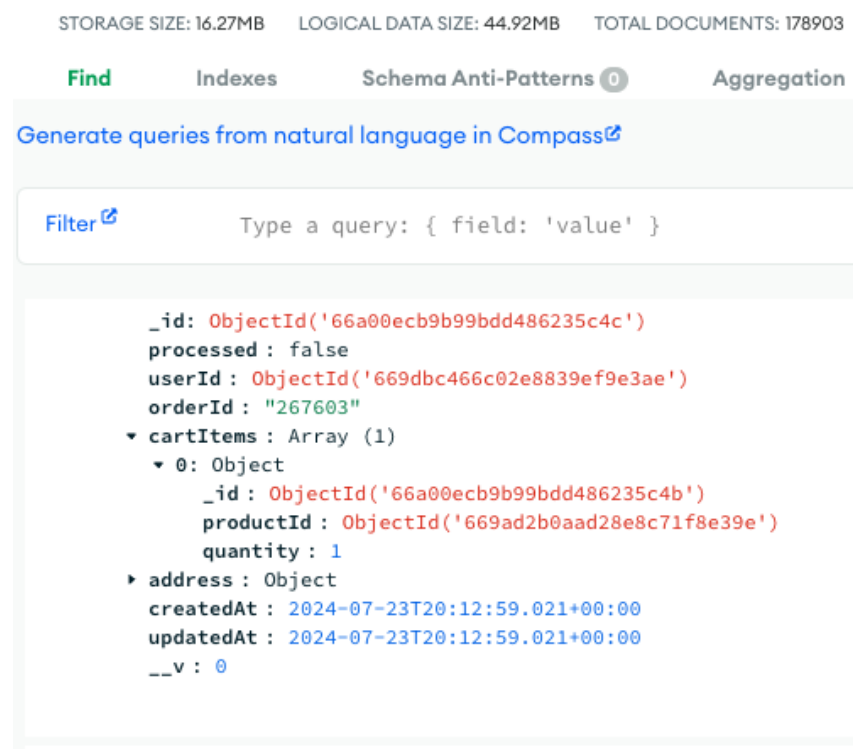


Рис. 2.6. Імпортовані замовлення в базу даних

Після завантаження даних в базу, сутність “Замовлення” має таку структуру:

Order – документ, що зберігає кошик користувача.

_id – унікальний ідентифікатор

cartItems – колекція об'єктів, що зберігають куплені товари, а саме: ідентифікатор товару та кількість.

userId – ідентифікатор користувача

processed – вказує чи замовлення оброблено адміністратором

payment – документ (), що зберігає інформацію про оплату

shipping – документ (), що зберігає інформацію про адресу доставки

total – загальна вартість замовлення

orderId – номер замовлення

createdAt – дата, коли замовлення було створено

updatedAt – дата, коли замовлення було редаговано

ShippingAddress – документ, що зберігає інформацію про адресу доставки.

country – назва країни доставки

city – місто доставки

address – адреса доставки

zip – поштовий індекс

Payment – документ, що зберігає інформацію про оплату.

_id – ідентифікатор платежу

amount – місто доставки

currency – адреса доставки

paid – поштовий індекс

payment_method – спосіб оплати

receipt_url – посилання на чек оплати

status – статус оплати

billing_details – деталі рахунку

2.2. Розробка рекомендаційної системи

Після аналізу методів та типів систем рекомендацій було реалізовано алгоритм колаборативної фільтрації на основі історії покупок користувача.

Як мову програмування було обрано Python, через ряд переваг:

- Багатий набір бібліотек і фреймворків : Python має численні бібліотеки, такі як scikit-learn, TensorFlow, Keras, Pandas, і Numpy, які спрощують розробку машинного навчання, обробку даних та аналіз.
- Зручність у використанні: Python відомий своїм простим і зрозумілим синтаксисом, що робить його доступним навіть для початківців.
- Велика спільнота: Python має велику та активну спільноту, яка постійно створює нові інструменти, бібліотеки та ресурси. Це забезпечує велику підтримку та доступ до знань, що корисно при вирішенні проблем.
- Кросплатформенність: Python є кросплатформенною мовою, що означає, що можна розгорнути свої рішення на різних операційних системах без значних змін в коді.
- Інтеграція з іншими технологіями: Python легко інтегрується з іншими мовами програмування та технологіями, що дозволяє використовувати його як частину більшого технологічного стека.
- Можливість обробки великих даних: Python має інструменти та бібліотеки для обробки великих обсягів даних, що є важливим аспектом для рекомендаційних систем, які часто потребують аналізу великих масивів даних.

В якості фреймворку для написання API був обраний FastAPI. FastAPI — це сучасний веб-фреймворк для створення API на Python. Він спрощує розробку високопродуктивних, швидких і простих у підтримці веб-додатків.

На цьому фреймворку було розроблено рекомендаційну систему, яка мала інтегруватися у вже існуючий додаток для електронної комерції розроблений у кваліфікаційній роботі на ступінь бакалавра. Цей додаток був розроблений з використанням стеку MERN. Клієнтська частина написана з використанням бібліотеки React.js, серверна частина на Node.js з використанням Express.js фреймворку. Як систему управління базами даних було вибрано MongoDB.

Щоб розробити рекомендаційну систему на основі історії покупок користувача, потрібно зробити такі кроки:

1. Завантажити дані історії покупок у Pandas DataFrame.
2. Перетворити дані у матрицю «користувач-елемент» : у цій матриці рядки представляють користувачів, а стовпці — елементи. Кожен запис у матриці представляє історію покупок користувача для товару. Наприклад, якщо користувач 1 придбав товари А і В, то запис у першому рядку та перших двох стовпцях матриці буде 1.
3. Обчислити подібності між користувачами. Обчислити косинусну матрицю подібності між продуктами, яка використовуватиметься для обчислення подібності між товарами, придбаними користувачем, і всіма іншими товарами.

```

purchase_counts = purchase_history.groupby(['userId', 'productId'], sort=False).size().unstack(fill_value=0)
sparse_purchase_counts = sparse.csr_matrix(purchase_counts)
cosine_similarities = cosine_similarity(sparse_purchase_counts.T)
print(cosine_similarities)

```

```

[[1.00000000e+00 3.12512103e-02 1.97127997e-02 1.56882083e-02
 1.13590008e-01 3.95997900e-02 3.21456398e-02 3.54496469e-02
 2.74242665e-02 3.38067533e-02 2.08576408e-02 1.57241042e-02
 1.65362264e-02 1.63948787e-02 1.95453538e-02 1.58796390e-02
 6.54062826e-02 7.67215051e-03 7.00018045e-03]]

```

Рис 2.7. Обчислення косинусної матриці

4. Обчислити історія покупок користувача та середню косинусну подібність між товарами, придбаними користувачем, і всіма іншими товарами.
5. Отримати індекси товарів, придбаних користувачем, і встановити оцінки схожості для цих товарів на 0.
6. Відсортувати елементи за показником подібності та повернути найперші елементи.
7. Видалити з рекомендованого списку товари, які користувач уже придбав.

Реалізувавши даний алгоритм, отримаємо код базової рекомендаційної системи (рис. 2.8). Це функція, що отримує ідентифікатор користувача та повертає 5 рекомендованих товари.

```
def recommend_items(user_id, n=5):
    unique_users = purchase_history["userId"].unique()

    order = np.where(unique_users == user_id)[0][0]

    user_history = sparse_purchase_counts[order].toarray().flatten()

    similarities = cosine_similarities.dot(user_history)

    purchased_indices = np.where(user_history > 0)[0]

    similarities[purchased_indices] = 0

    recommended_indices = np.argsort(similarities)[::-1][:100]

    recommended_items = list(purchase_counts.columns[recommended_indices])

    purchased_items = list(purchase_counts.columns[purchase_counts.loc[user_id] > 0])
    recommended_items = [item for item in recommended_items if item not in purchased_items]

    return recommended_items[:n]

print(recommend_items(user_id))
```

→ ['669ad2b0aad28e8c71f8e39e', '669ad2b0aad28e8c71f8e3a0', '669ad2b0aad28e8c71f8e3a1', '669ad2b0aad28e8c71f8e3

Рис 2.8. Базовий алгоритм рекомендаційної системи

Для покращення рекомендаційної системи, можемо також врахувати товари, що були додані в кошик користувача, в такому випадку рекомендації будуть більш дієві, так як будуть враховувати інтереси користувача прямо під час оформлення замовлення. Для цього отримуємо з бази даних інформацію про кошик користувача та додамо товари до загального списку замовлень.

```
cart = request.app.database["carts"].find_one({"userId": ObjectId(id)})

if(cart and 'cartItems' in cart and len(cart['cartItems']) > 0):
    extracted = extract_cart_items(cart)
    purchase_history = pd.concat([purchase_history, pd.DataFrame(extracted)], ignore_index=True)
```

Рис 2.9. Враховуємо товари в кошику

У випадку, якщо у користувача немає куплених товарів раніше, можемо йому рекомендувати найпопулярніші товари.

```
def return_most_popular(purchase_history):
    return purchase_history['productId'].value_counts()[:5].index.tolist()

if(unique_user_orders[0].size == 0):
    return return_most_popular(purchase_history)
```

Рис 2.10. Рекомендація найпопулярніших продуктів

Наступним кроком, розгорнемо API для роботи з рекомендаційною системою, використавши Fast API. Запити з клієнту будуть приходити на сервер інтернет магазину, що був розроблений раніше на Node.js. З серверу інтернет магазину буде йти запит на сервер рекомендаційної системи, як результат отримаємо мікросерверну архітектуру додатку.

Висновок до другого розділу:

У цьому розділі було проведено аналіз вхідних даних, виявлено продукти, що купуються разом найчастіше, та на основі цього розроблено рекомендаційну систему що повертає рекомендовані товари для користувача на основі історії його покупок. Обґрунтовано вибір інструментів для розробки, розроблено та описано базовий алгоритм для рекомендаційної системи, який використовує метод колаборативної фільтрації та як функцію схожості використано функцію косинусної схожості. Також, було доопрацьовано алгоритм, та розгорнуто API для роботи з рекомендаційною системою.

РОЗДІЛ 3. ДОСЛІДЖЕННЯ ПРАЦЕЗДАТНОСТІ ВЕБ-ДОДАТКУ

3.1. Архітектура програмного комплексу та його розгортання

Архітектуру програмного продукту та його розгортання зазвичай ілюструють за допомогою діаграми розгортання.

Діаграма розгортання — це діаграма, на якій показуються обчислювальні вузли під час виконання програми, а також компоненти та об'єкти, які працюють на цих вузлах. Компоненти являють собою робочі екземпляри модулів коду.

Компоненти, що не функціонують під час виконання програми, на таких діаграмах не відображаються. Замість цього їх можна показати на діаграмах компонентів. Діаграма розгортання показує робочі екземпляри компонентів, а діаграма компонентів, відповідно, показує взаємозв'язки між типами компонентів.

Існує два типи вузлів:

- Вузол пристрою;
- Вузол середовища виконання.

Вузли пристроїв - це фізичні обчислювальні ресурси зі своєю пам'яттю і сервісами для виконання програмного забезпечення, такі як звичайні ПК, мобільні телефони.

Вузол середовища виконання - це програмний обчислювальний ресурс, який працює всередині зовнішнього вузла і який надає собою сервіс, який виконує інші виконувані програмні елементи [6].

Діаграму розгортання додатку зображено на рисунку 3.1. Діаграма розгортання має наступну структуру:

Вузли:

Клієнт - обчислювальна машина користувача

Сервер - сервер для обробки запитів користувачів

Сервер бази даних

Сервер рекомендаційної системи

Компоненти вузла «Клієнт»:

Браузер

React.js – бібліотека, яка будує користувацький інтерфейс.

Компоненти вузла Server:

Node.js - це JavaScript-оточення побудоване на JavaScript-рушієві

Компоненти вузла Database Server:

MongoDb- документо-орієнтована система керування базами даних

Компоненти вузла Recommendation Server:

Fast API - веб-фреймворк для створення API на Python.

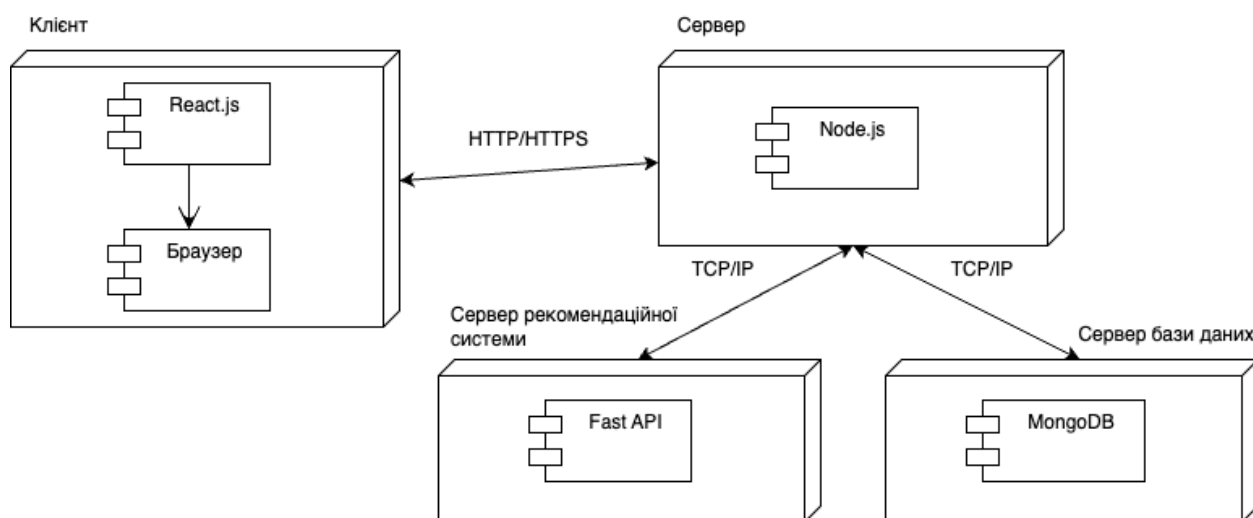


Рисунок 3.1. Діаграма розгортання

3.2. Оцінка та аналіз якості моделі.

Оцінка точності рекомендаційної системи на основі колаборативної фільтрації може бути здійснена за допомогою різних метрик, таких як точність (Precision), повнота (Recall), F1-міра (F1 Score), тощо. Вибір метрики залежить від конкретної задачі та контексту використання системи рекомендацій [16].

Точність (Precision)

Точність вимірює частку правильно передбачених продуктів серед усіх передбачених.

Формула обчислення точності:

$$P = \frac{N_{rs}}{N_s}, \quad (3.1)$$

де:

N_{rs} — кількість правильно передбачених продуктів,

N_s — кількість всіх передбачених продуктів.

Повнота (Recall)

Повнота вимірює частку правильно передбачених продуктів серед усіх куплених користувачем.

Формула обчислення повноти:

$$R = \frac{N_{rs}}{N_r}, \quad (3.2)$$

де:

N_{rs} — кількість правильно передбачених продуктів.

N_r — кількість всіх куплених продуктів користувачем.

F1-міра (F1 Score)

F1-міра є середнім гармонійним між точністю і повнотою. Вона забезпечує баланс між ними і є корисною, коли важливо як уникати хибних позитивних, так і не пропускати справжні позитивні випадки.

Формула обчислення F міри:

$$F_1 = \frac{2PR}{P+R}, \quad (3.3)$$

де:

P — Точність

R — Повнота

Для обчислень цих метрик, нам потрібно поділити дані на тестові та навчальні. Розділимо їх в пропорції 90% навчальні та 10% тестові.

```
def evaluate_recommendations(test_data, recommendations_func):
    precisions = []
    recalls = []
    f1s = []

    for user_id in user_ids:
        true_purchases = test_data[test_data['userId'] == user_id]['productId'].unique()

        if(len(true_purchases) == 0):
            continue

        recommended_items = recommendations_func(user_id)

        precision = np.sum(np.isin(recommended_items, true_purchases)) / len(recommended_items)
        recall = np.sum(np.isin(recommended_items, true_purchases)) / len(true_purchases)
        f1 = 2 * (precision * recall) / (precision + recall) if (precision + recall) != 0 else 0

        precisions.append(precision)
        recalls.append(recall)
        f1s.append(f1)

    avg_precision = np.mean(precisions)
    avg_recall = np.mean(recalls)
    avg_f1 = np.mean(f1s)

    return avg_precision, avg_recall, avg_f1
```

Рис 3.2. Функція для обчислення точності рекомендацій

Далі, обчислимо модель схожості на навчальних даних. Наступним кроком, потрібно отримати рекомендації для користувачів з тестового набору, та порівняти рекомендації з реальними покупками користувача для кожної метрики відповідно до формули.

Врахуємо той факт, що максимальна кількість замовлень унікального товару на одного користувача не перевищує 3, а кількість рекомендацій, що пропонує алгоритм - 5, що продиктовано інтерфейсом програми, а тому як загальну кількість рекомендованих товарів N_s будемо використовувати 3.

```
Mean Avarage Precision: 0.3107638888888889
Mean Avarage Recall: 0.6041666666666666
Mean Avarage F1 Score: 0.40856481481481477
```

Рис 3.3. Результат обчислення точності

Отримані результати точності можна вважати високими, так як користувач купує товари в залежності від своїх потреб, а не засновуючись на даних про покупки інших користувачів. На відміну від задач кластеризації чи класифікації, де ми точно можемо сказати до якого кластеру чи класу належить сутність, в задачі рекомендації продуктів ми не можемо гарантувати, що саме цей набір продуктів буде куплений користувачем, щоб порівняти з рекомендованим.

Отриманні дані можуть використанні для порівнянь алгоритмів та їх оптимізації.

Також, можемо визначити точність (Precision) та повноту (Recall) для кожного окремого продукту, та вивести інформацію в графічному вигляді. Для обчислення точності рекомендації продукту потрібно суму кількості правильних рекомендацій для всіх користувачів поділити на суму загальної кількості запропонованих рекомендацій для всіх користувачів. Для обчислення повноти потрібно суму кількості правильних рекомендацій поділити на суму кількості куплених продуктів всіма користувачами.

```
for user_id in user_ids:
    true_purchases = test_data[test_data['userId'] == user_id]['productId'].unique()

    if(len(true_purchases) == 0):
        continue

    recommended_items = recommendations_func(user_id)

    for product in true_purchases:
        if product not in product_metrics:
            product_metrics[product] = {'correct': 0, 'recommended': 0, 'total': 0}

        if product in recommended_items:
            product_metrics[product]['correct'] += 1

        product_metrics[product]['total'] += 1
        product_metrics[product]['recommended'] += 3

product_avg_precisions = {
    product: (values['correct'] / values['recommended']) if values['recommended'] > 0 else 0
    for product, values in product_metrics.items()
}
product_avg_recall = {
    product: (values['correct'] / values['total']) if values['total'] > 0 else 0
    for product, values in product_metrics.items()
}
```

Рис 3.4. Визначення метрик для продукту

Проте, одна лише хороша точність рекомендацій не забезпечує користувачам систем рекомендацій ефективний і задовільний досвід. Системи рекомендацій повинні надавати не тільки точність, але й корисність.

Наприклад, рекомендаційна система може досягти високої точності, тільки обчислюючи передбачення для легко прогнозованих елементів, але бувають

випадки, що саме для цих елементів користувачам найменш імовірно потрібні передбачення.

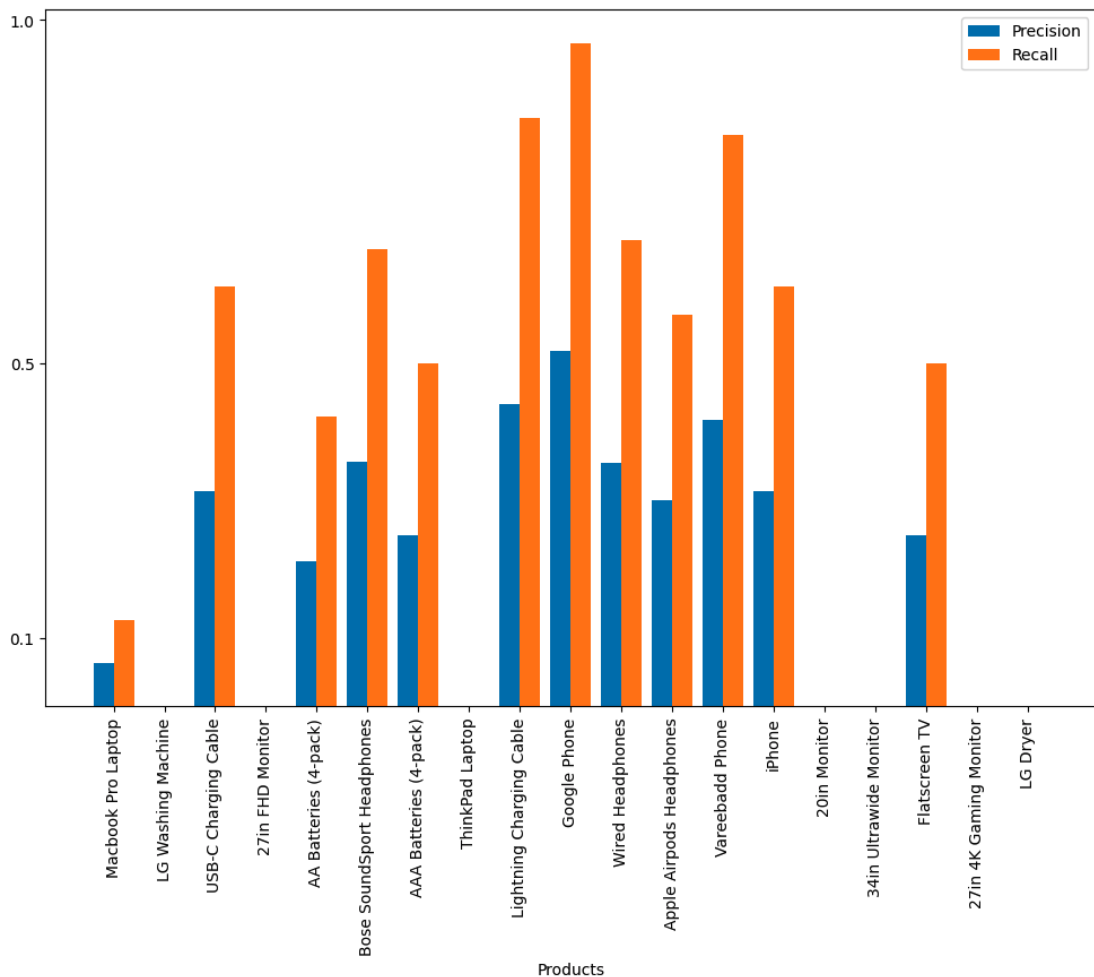


Рис 3.5. Метрики по кожному продукту

Враховуючи, що ефективність системи рекомендацій повинна оцінюватися з урахуванням конкретних завдань користувача та контексту домену, ми можемо поглибити аргумент оцінки ефективності рекомендаційної системи на користь виходу за межі тільки точності.

3.3. Перевірка працездатності додатку

Перевірка на працездатність – фінальний та найважливіший етап під час розробки програмного продукту.

Тестування визначається як процес перевірки коректності програми в реальному часі за допомогою тестових даних. Під час тестування виявляються

проблеми: збої та дефекти, що викликають порушення роботи програми, збої як небажані ситуації, помилки як наслідки збоїв тощо. Основне поняття тестування – тест, що проводиться за певних умов та з використанням певних наборів даних.

За об'єктом тестування розрізняють такі види тестування ПЗ:

- функціональне тестування (functional testing);
- тестування продуктивності (performance testing);
- стрес-тестування (stress testing);
- тестування зручності використання (usability testing);
- тестування інтерфейсу користувача (UI testing);
- тестування безпеки (security testing);
- тестування сумісності (compatibility testing).
- інші.

Залежно від цілей, види тестування умовно поділяють на такі типи:

- функціональне тестування (functional);
- нефункціональне тестування (non-functional);
- тестування, пов'язане зі змінами.

Functional testing (Функціональне тестування) — передбачає аналіз функціональних характеристик додатка та перевірку на відповідність між фактичною поведінкою реалізованих функцій і очікуваною поведінкою відповідно до специфікацій і бізнес-вимог. Функціональне тестування імітує реальне використання системи.

Відкриємо клієнтський додаток на головній сторінці. Можемо бачити блок з рекомендаціями, який може містити 5 або 15 рекомендованих продуктів показаних в каруселі.

Блок з рекомендаціями було додано на головну сторінку, а також на сторінку кошику.

Один з недоліків колаборативної фільтрації це проблема холодного старту. Один із варіантів вирішення цієї проблеми це повертати універсальні рекомендації, наприклад найпопулярніші товари. Саме таким чином було

вирішено проблему холодного старту в додатку. Тому, для незареєстрованого користувача, або для користувача який не мав замовлень, повертаються 15 найпопулярніших продуктів.

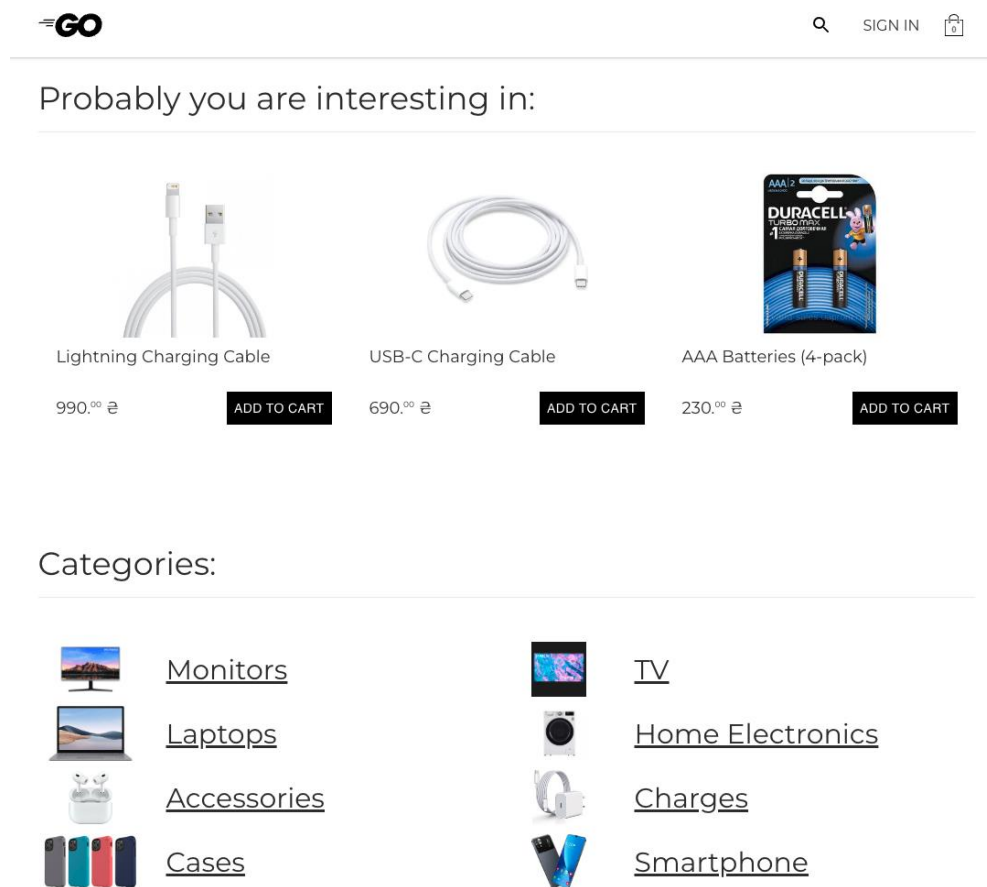


Рис 3.6. Рекомендації на головній сторінці

Зареєструємо нового користувача та увійдемо в обліковий запис. Рекомендації не змінились, так як користувач не мав замовлень до цього.

Алгоритм рекомендації також враховує продукти що були додані в кошик, тому для перевірки додамо в кошик продукт “iPhone” та перейдемо в кошик.

В списку рекомендацій відображені такі товари, як “Lightning Charging Cable”, “Wired Headphones”, “Apple Airpods Headphones”, що є логічним для товару компанії Apple. Також, можемо порівняти запропоновані рекомендації з тепловою картою спільних появ продуктів (рис. 2.4).

Можемо бачити, що список рекомендацій повністю співпадає з товарами

на тепловій карті, що були замовлені одним користувачем найчастіше.

Items in you cart:

Product	Description	Quantity	Price	Remove
	iPhone	< 1 >	45800 ₺	×

TOTAL: 45800 ₺

CHECKOUT

Recommendations for you:



Lightning Charging Cable

990.00 ₺

ADD TO CART



Wired Headphones

210.00 ₺

ADD TO CART



Apple AirPods Headphones

5660.00 ₺

ADD TO CART

Рис 3.7. Рекомендації на для товару iPhone

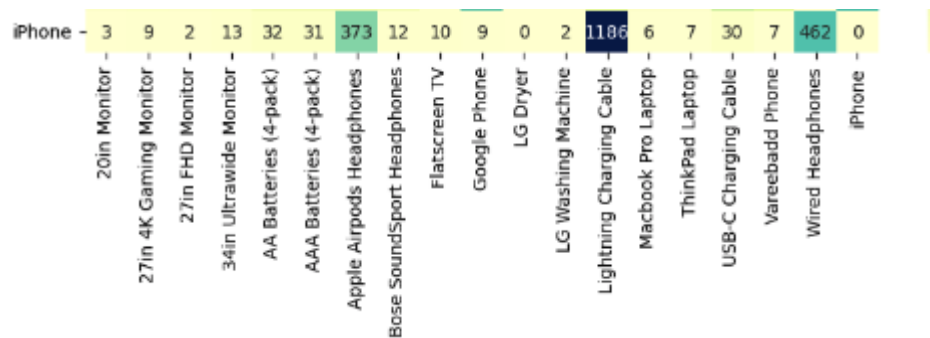


Рис. 3.8. Теплова карта спільних появ продуктів із товаром iPhone

Перевіримо рекомендації для іншого продукту, очистимо кошик да додамо товар “Google Phone”. Для цього продукту відповідні зовсім інші рекомендації: “Lightning Charging Cable” відсутній в списку, так як Google Phone використовує “USB-C Charging Cable” зарядний пристрій, який ми бачимо першим в списку рекомендацій. Правильність рекомендацій ми також можемо перевірити відносно теплової карти в 2 розділі (рис. 2.4), де разом з товаром Google Phone

найбільшу кількість покупок мали товари “USB-C Charging Cable”, “Wired Headphones” та “Bose SoundSport Headphones”.

Items in you cart:

Product	Description	Quantity	Price	Remove
	Google Phone	< 1 >	32100 ₾	×
TOTAL: 32100 ₾				CHECKOUT

Recommendations for you:

		
USB-C Charging Cable	Wired Headphones	Bose SoundSport Headphones
690.00 ₾ ADD TO CART	210.00 ₾ ADD TO CART	2320.00 ₾ ADD TO CART

Рис. 3.9. Рекомендації на для товару Google Phone

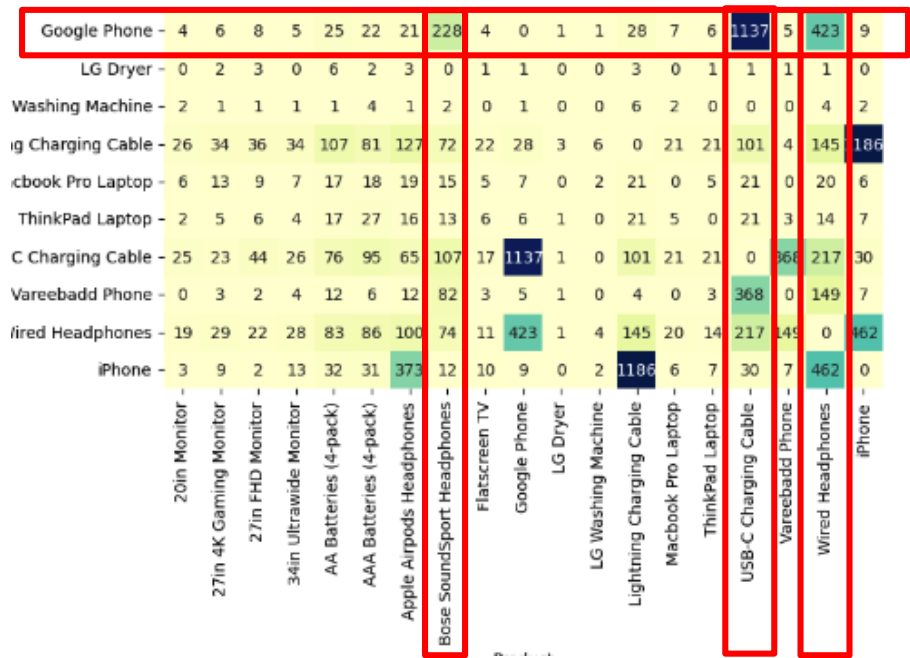


Рис. 3.10. Теплова карта спільних появ продуктів із товаром Google Phone

Додавши в кошик декілька товарів, ми побачимо список спільних рекомендацій для всіх товарів, доданих в кошик. Як бачимо на рисунку 3.11, в рекомендаціях є товари “Lightning Charging Cable” та “USB-C Charging Cable” одночасно.

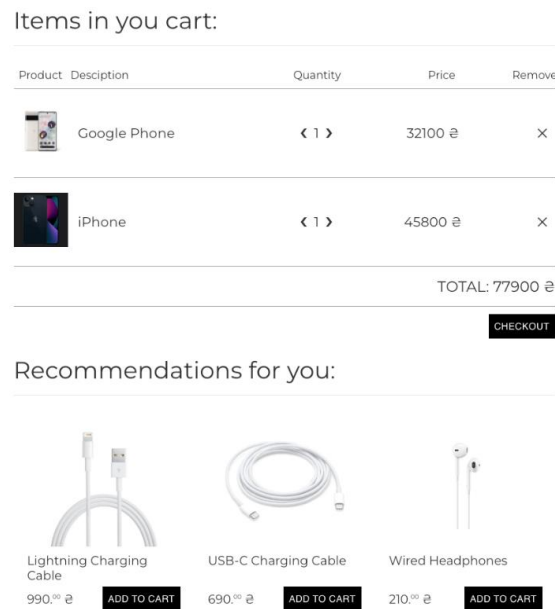


Рис. 3.11. Рекомендації для декількох товарів

Висновки до третього розділу:

В даному розділі було описано інтерфейс блоку рекомендацій, що інтегрувався в готовий інтернет магазин. Розглянуто архітектуру програмного продукту та його розгортання за допомогою діаграми розгортання.

Було проведено оцінку точності рекомендаційної системи за допомогою різних метрик, таких як точність (Precision), повнота (Recall), F1-міра (F1 Score). Також, була визначена точність та повнота для кожного окремого продукту, та отримана інформація виведена в графічному вигляді.

Було описано та наведено приклади роботи рекомендаційної системи, порівняно список рекомендацій на прикладі декількох різних товарів, отримані результати були звірено з тепловою картою появи спільних продуктів, для аргументації правильності рекомендованих товарів. В розділі основна увага зосереджена на перевірці працездатності рекомендаційної системи.

ВИСНОВКИ

Під час роботи над кваліфікаційною роботою магістра було проаналізовано актуальність теми аналізу даних електронної комерції та застосування рекомендаційних систем. Розглянуто важливість аналізу продажів у роздрібній торгівлі, зокрема для інтернет-магазинів.

Розглянуто різні методи аналізу даних, описано, що таке рекомендаційна система та як вона працює, розглянуто різні підходи до методу колаборативної фільтрації, такі як Item-Item, User-user та гібридні, розглянуто функції схожості: косинусна та коефіцієнт Пірсона.

Було проведено аналіз вхідних даних, виявлено продукти, що купуються разом найчастіше, та на основі цього розроблено рекомендаційну систему що повертає рекомендовані товари для користувача на основі історії його покупок. Обґрунтовано вибір інструментів для розробки, розроблено та описано базовий алгоритм для рекомендаційної системи, який використовує метод колаборативної фільтрації та як функцію схожості використано функцію косинусної схожості. Також, було доопрацьовано алгоритм, та розгорнуто API для роботи з рекомендаційною системою.

Було описано інтерфейс блоку рекомендацій, що інтегрувався в готовий інтернет магазин. Розглянуто архітектуру програмного продукту та його розгортання за допомогою діаграми розгортання.

Було проведено оцінку точності рекомендаційної системи за допомогою різних метрик, таких як точність (Precision), повнота (Recall), F1-міра (F1 Score).

Також, була визначена точність та повнота для кожного окремого продукту, та отримана інформація виведена в графічному вигляді.

Було описано та наведено приклади роботи рекомендаційної системи, порівняно список рекомендацій на прикладі декількох різних товарів, отримані результати були звірено з тепловою картою появи спільних продуктів, для аргументації правильності рекомендованих товарів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Moroney L. AI and Machine Learning for Coders: A Programmer's Guide to Artificial Intelligence / Laurence Moroney., 2020.
2. Gedeck P. Practical Statistics for Data Scientists: 50+ Essential Concepts Using R and Python / P. Gedeck, A. Bruce, P. Bruce., 2020. – 368 с.
3. Geron A. Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems / Aurelien Geron., 2023. – 850 с.
4. Mastrodomenico R. The Python Book / Rob Mastrodomenico., 2022. – 272 с.
5. Kumar P. Recommender Systems: Algorithms and Applications / Pavan Kumar., 2021. – 230 с.
6. Falk K. Practical Recommender Systems / Kim Falk., 2019. – 430 с.
7. Bischof B. Building Recommendation Systems in Python and JAX: Hands-On Production Systems at Scale / B. Bischof, H. Yee., 2024.
8. Kane F. Building Recommender Systems with Machine Learning and AI / Frank Kane., 2021. – 544 с. – (2).
9. Berkovsky S. Collaborative Recommendations: Algorithms, Practical Challenges And Applications / S. Berkovsky, I. Cantador, D. Tikk., 2019. – 736 с.
10. Chollet F. Deep Learning with Python / Francois Chollet., 2021. – 504 с.
11. Мистецтво аналізу продажів [Електронний ресурс]. – 2023. – Режим доступу до ресурсу: <https://datawiz.io/uk/blog/the-art-of-sales-analysis-basic-guide-for-retail-chains>.
12. Аналітика інтернет-магазину [Електронний ресурс]. – 2022. – Режим доступу до ресурсу: <https://horoshop.ua/ua/blog/online-store-analytics/>.
13. Основи статистики та аналізу даних [Електронний ресурс] – Режим доступу до ресурсу: <https://socialdata.org.ua/manual/manual4/>.
14. Полурезов Д. Від концепції до реалізації: як побудувати ефективну рекомендаційну систему на основі машинного навчання [Електронний

ресурс] / Дмитро Полурезов. – 2024. – Режим доступа до ресурсу:
<https://dou.ua/forums/topic/47504/>.

- 15.Lomsadze S. Building a Machine Learning Model for Product Recommendations Using Customer Purchase Data [Электронный ресурс] / Sonya Lomsadze. – 2023. – Режим доступа до ресурсу:
<https://salomelomsadze.medium.com/building-a-machine-learning-model-for-product-recommendations-using-customer-purchase-data-95bd86da7350>.
- 16.Herlocker J. Evaluating Collaborative Filtering Recommender Systems [Электронный ресурс] / Jonathan Herlocker – Режим доступа до ресурсу:
<https://pzs.dstu.dp.ua/DataMining/recom/bibl/10.1.1.97.5270.pdf>.

ДОДАТКИ

Перегляд завантажених даних

```
import pandas as pd

import numpy as np

from sklearn.metrics.pairwise import cosine_similarity

from scipy import sparse


data = pd.read_csv('./drive/MyDrive/df1.csv')

data.info()

data.head()


product_counts = data['productId'].value_counts()

product_counts.plot(kind='bar')

plt.show()
```

Очищення даних від нульових значень

```
import pandas as pd

data = pd.read_csv('./drive/MyDrive/df1.csv')


data_cleaned = data.dropna()

missing_values = data_cleaned.isnull().sum()

print(missing_values)
```

Аналіз спільної появи продуктів та виведення теплової карти

```
import pandas as pd

import numpy as np
```

```

from sklearn.cluster import KMeans

import matplotlib.pyplot as plt

import seaborn as sns

product_matrix = pd.crosstab(index=data['Order ID'], columns=data['Product'])

co_occurrence_matrix = np.dot(product_matrix.T, product_matrix)

np.fill_diagonal(co_occurrence_matrix, 0)

co_occurrence_df = pd.DataFrame(co_occurrence_matrix, index=product_matrix.columns,
                                columns=product_matrix.columns)

num_clusters = 2

kmeans_products = KMeans(n_clusters=num_clusters, random_state=42)

kmeans_products.fit(co_occurrence_df)

product_labels = kmeans_products.labels_

product_cluster_df = pd.DataFrame({
    'Product': product_matrix.columns,
    'Cluster': product_labels
}).sort_values(by='Cluster')

plt.figure(figsize=(10, 8))

sns.heatmap(co_occurrence_df, cmap="YlGnBu", annot=True, fmt='d')

plt.title('Product Co-Occurrence Matrix')

plt.show()

```

Виведення графу спільної появи продуктів

```

G = nx.Graph()

for i, product in enumerate(co_occurrence_df.index):
    G.add_node(product, cluster=product_labels[i])

```

```

for i, product1 in enumerate(co_occurrence_df.index):
    for j, product2 in enumerate(co_occurrence_df.columns):
        weight = co_occurrence_df.at[product1, product2]
        if weight > 0:
            G.add_edge(product1, product2, weight=weight)

edges_to_keep = [(u, v) for u, v, d in G.edges(data=True) if d['weight'] > 10]
G_filtered = G.edge_subgraph(edges_to_keep)

colors = [G_filtered.nodes[node]['cluster'] for node in G_filtered.nodes]
colors2 = ['pink' if x == 0 else 'yellow' if x == 1 else x for x in colors]

plt.figure(figsize=(7, 5))
nx.draw_networkx(
    G_filtered,

    node_color=colors2,
    with_labels=True,
    node_size=500,
    font_size=8,
    font_color='black',
    edge_color='gray',

)

plt.show()

```

Оцінка точності алгоритму

```
from sklearn.metrics import average_precision_score

from sklearn.metrics import ndcg_score


user_ids = purchase_history['userId'].unique()


def evaluate_recommendations(test_data, recommendations_func):

    precisions = []

    recalls = []

    f1s = []

    users = []

    for user_id in user_ids:

        true_purchases = test_data[test_data['userId'] == user_id]['productId'].unique()

        if(len(true_purchases) == 0):

            continue

        recommended_items = recommendations_func(user_id)

        precision = np.sum(np.isin(recommended_items, true_purchases)) / (len(recommended_items) - 3)

        recall = np.sum(np.isin(recommended_items, true_purchases)) / len(true_purchases)

        f1 = 2 * (precision * recall) / (precision + recall) if (precision + recall) != 0 else 0

        precisions.append(precision)

        recalls.append(recall)
```

```

f1s.append(f1)

users.append(user_id)


avg_precision = np.mean(precisions)
avg_recall = np.mean(recalls)
avg_f1 = np.mean(f1s)


return avg_precision, avg_recall, avg_f1, precisions, recalls, users


avg_precision, avg_recall, avg_f1, precisions, recalls, users = evaluate_recommendations(test_data,
recommend_items)

print("Mean Avarage Precision:", avg_precision)

print("Mean Avarage Recall:", avg_recall)

print("Mean Avarage F1 Score:", avg_f1)

```

Обчислення точності рекомендацій для кожного продукту

```

user_ids = purchase_history['userId'].unique()


def get_precisions_for_products(product_list, product_avg_precisions):
    return [product_avg_precisions.get(product, 0) for product in product_list]


def evaluate_recommendations(test_data, recommendations_func):
    product_metrics = {}


    for user_id in user_ids:
        true_purchases = test_data[test_data['userId'] == user_id]['productId'].unique()

```

```

if(len(true_purchases) == 0):

    continue

recommended_items = recommendations_func(user_id)

for product in true_purchases:
    if product not in product_metrics:
        product_metrics[product] = {'correct': 0, 'recommended': 0, 'total': 0}

    if product in recommended_items:
        product_metrics[product]['correct'] += 1

        product_metrics[product]['total'] += 1
        product_metrics[product]['recommended'] += 3

product_avg_precisions = {
    product: (values['correct'] / values['recommended']) if values['recommended'] > 0 else 0
    for product, values in product_metrics.items()
}

product_avg_recall = {
    product: (values['correct'] / values['total']) if values['total'] > 0 else 0
    for product, values in product_metrics.items()
}

return product_avg_precisions, product_avg_recall

product_avg_precisions, product_avg_recall = evaluate_recommendations(test_data, recommend_items)
product_precisions = get_precisions_for_products(listOfIds, product_avg_precisions)

```

```
product_recall = get_precisions_for_products(listOfIds, product_avg_recall)
```

Виведення діаграми точності рекомендацій для продукту

```
x = np.arange(len(products)) # the label locations
width = 0.4 # width of bars
print(product_precisions)
# Create a bar plot
fig, ax = plt.subplots(figsize=(12, 8))
bars1 = ax.bar(x - width/2, product_precisions, width, label='Precision')
bars2 = ax.bar(x + width/2, product_recall, width, label='Recall')

# Add labels, title, and legend
ax.set_xlabel('Products')
ax.set_ylabel('Values')
ax.set_xticks(x)
ax.set_xticklabels(products)
ax.legend()
plt.xticks(rotation=90)
ticks = [0.1, 0.5, 1]
plt.yscale('linear')
plt.yticks(ticks, [f'{tick:.1f}' for tick in ticks])
# Show the plot
plt.show()
```

Код серверу рекомендацій.

Файл main.py

```
from fastapi import FastAPI, Request
from dotenv import dotenv_values
from pymongo import MongoClient
from routes import router
from db import mongodb_client, database

config = dotenv_values(".env")

app = FastAPI()

@app.on_event("startup")
def startup_db_client():
    app.mongodb_client = mongodb_client
    app.database = database
    print("Connected to the MongoDB database!")

@app.on_event("shutdown")
def shutdown_db_client():
    app.mongodb_client.close()

app.include_router(router, tags=["api"], prefix="/api")
```

A

```
from fastapi import APIRouter, Body, Request, Response, HTTPException, status
from fastapi.encoders import jsonable_encoder
from typing import List
from bson import json_util
import pandas as pd
import json
from scipy import sparse
from sklearn.metrics.pairwise import cosine_similarity
import numpy as np
from bson.objectid import ObjectId
from db import database
from scv import save_to_scv, read_from_csv

router = APIRouter()
```

```

def extract_cart_items(cart_document):
    cart_items = cart_document['cartItems']
    user_id = cart_document['userId']

    # Prepare data for DataFrame
    data = [{ 'productId': str(item['productId']), 'userId': str(user_id)} for item in cart_items]
    return data

def return_most_popular(purchase_history):
    return purchase_history['productId'].value_counts()[:15].index.tolist()

@router.get("/recomm/{id}")
def get_recommendations(id: str, request: Request):
    purchase_history = read_from_csv()

    # return most popular if userId is missing
    if(id == "undefined"):
        return return_most_popular(purchase_history)

    # added items from cart if exist
    cart = request.app.database["carts"].find_one({ "userId": ObjectId(id)})

    if(cart and 'cartItems' in cart and len(cart['cartItems']) > 0):
        extracted = extract_cart_items(cart)
        purchase_history = pd.concat([purchase_history, pd.DataFrame(extracted)], ignore_index=True)

    purchase_counts = purchase_history.groupby(['userId', 'productId'], sort=False).size().unstack(fill_value=0)

    sparse_purchase_counts = sparse.csr_matrix(purchase_counts)

    cosine_similarities = cosine_similarity(sparse_purchase_counts.T)

    unique_users_orders = purchase_history["userId"].unique()

    unique_user_orders = np.where(unique_users_orders == id)

    #id there is no data oin order - user dont have orders
    if(unique_user_orders[0].size == 0):
        return return_most_popular(purchase_history)

    userIndex = unique_user_orders[0][0]

    user_history = sparse_purchase_counts[userIndex].toarray().flatten()

```

```
similarities = cosine_similarities.dot(user_history)

purchased_indices = np.where(user_history > 0)[0]

similarities[purchased_indices] = 0

recommended_indices = np.argsort(similarities)[::-1][:5]
recommended_items = list(purchase_counts.columns[recommended_indices])

purchased_items = list(purchase_counts.columns[purchase_counts.loc[id] > 0])
recommended_items = [item for item in recommended_items if item not in purchased_items]

return recommended_items
```